Amendments to the Claims:

This listing of claims will replace all prior versions, and listings, of claims in the application:

Listing of Claims:

Claims 1-104 (cancelled).

105. (currently amended) A ~~data processing~~ computer system for automatically generating a ~~global~~ simulation model of a configuration of software simulation elements, comprising:

storage means for storing a plurality of software simulation elements, said [[a]] plurality of ~~said~~ software simulation elements provided with ~~mutually connected by~~ inter working connections so as to constitute the ~~global~~ simulation model of an architecture, wherein the ~~global~~ simulation model comprises models of integrated circuits for the realization of a machine that conforms to a functional specification of a configuration defined in a configuration definition file; and ~~by a user,~~

a data processing system comprising execution means provided with ~~a Configurator further comprising~~ configuration means that comprises:

means for creating a simulation of wiring by executing stored regular expressions, ~~and~~

means for using the configuration definition file, a component and connection rule table, and a connection coherency rule table;[[,]] wherein the component and connection rule table and the connection coherency rule table are

~~being~~ written in a high level language (HLL), and the component and connection rule table ~~describing~~ describes properties of software components for simulating ~~the~~ at least one of the integrated circuits, ~~circuit~~, and

~~the connection coherency rule table being written in [[a]] high level language, and~~

means for instantiating components resulting from the configuration definition file, and

an HLL code generator that combines ~~the~~ parameters of the components with ~~the~~ connection ~~roles~~ rules of the component and connection rule table.

106. (previously presented) A system according to claim 105, wherein the components comprise Active Components, Monitoring and Verification Blocks, Intermediate Blocks, System Blocks, and Global Blocks.

107. (currently amended) A system according to claim 106, further comprising means to perform a conformity check of the connections by comparing ~~the~~ an instance connection table with a table of coherency rules for the physical connections between the models chosen from the blocks constituting the ~~global~~ model.

108. (currently amended) A system according to claim 107, further comprising means to compare the instance connection table to the connection coherency rule table to detect any incompatible connections between the ends of the connections between blocks, and incases where an incompatible connection is detected, the ~~data processing~~ system is configured to specify and add an adapter component (Intermediate Block) to the instance connection table, said adapter component being inserted into the incompatible connection between the components.

109. (currently amended) A system according to claim 108, wherein the component and connection rule table includes ~~the~~ properties of the components and contains ~~global~~ parameters common to all of the component types and exists in the form of a table distributed into one or more associative tables, wherein the entries are names designating all of the possible models for the same component.

110. (currently amended) A system according to claim 109, wherein the associative tables are adapted to contain a description either in the form of parameter sets or in the form of references to procedures that generate the required values, ~~wherein~~ and the entries of these associative tables comprise names designating all of the possible models for the same component[[,]] and form a character string containing predetermined special identifiers that are replaced by values calculated by the ~~Configurator~~ configuration means.

111. (previously presented) A system according to claim 110, wherein at least three selectors indicate the instance to be used, and in which the following selectors are transmitted as parameters to a constructor of an HLL object:

a first selector indicating a current instance (item);

a second selector specifying the current instance connected to the other end of the port; and

a third selector indicating a composite instance corresponding to an active Component containing an observation port.

112. (currently amended) A system according to claim 105, wherein the ~~Configurator~~ configuration means further comprises:

one or more connection coherency rule tables representing the rules for interconnecting the components and for inserting intermediate components;

one or more component and connection rule tables representing the system-level connection rules and the rules for generating connections between the signals; and

one or more source file formatting tables representing the rules for generating instances of HLL ~~HLL-type~~ objects.

113. (currently amended) A system according to claim 105, wherein the ~~Configurator~~ configuration means further comprises:

an HLL base class uniquely identifying each object instantiated and polling the configuration;

means for generating and automatically instantiating System Blocks;

means for using tables to associate the signals connected together under a unique name of the connecting wires; and

means for using a formatting table to generate ~~HDL-type and HLL-type~~ hardware description language (HDL) and HLL source files.

114. (currently amended) A system according to claim 105, wherein the system is configured to receive from an operator a functional specification of the configuration in a high level language, and to complete the functional specification with the components in a language having a level lower than said high level language.

115. (currently amended) A system according to claim 105, wherein the following entries in a hash define a Component Type and correlate each Component Type to [[a]] the hash, said hash comprising the following:

a first entry comprising a name of ~~the~~ a hardware description language (HDL) module of ~~the~~ a component and a name of a corresponding source file; and

a second entry comprising a definition of a method for selecting the signals that are part of a Port, said definition comprising a set of entries indexed by ~~the~~ a name of the Port;

wherein the ~~Configurator~~ configuration means is configured to associate each said Port name with a table of regular expressions and a pointer to a signal connection procedure that controls the application of the expressions to the names of the signals of the interface of the component.

116. (currently amended) A system according to claim 115, wherein said Component Type comprises one or more Active Components having a generic structure that includes a containing Block that contains ~~containing the~~ an HDL Block including an HDL description and a Block in HLL that provides ~~the~~ access paths to ~~the~~ HDL resources[[,]] and a description of the containing block in the high level language; ~~HLL;~~

wherein the set of signals of the HDL Block constitutes ~~block constitute the~~ an interface of the containing Block, formed by Ports, which are arbitrary logical selections of the signals of an interface, and also formed by interface adapters which are the software parts that handle, in each Port, the two-way communication between the parts in high level language ~~HLL~~ and those in hardware description language ~~HDL,~~ the interface adapters being selected by the ~~Configurator~~ configuration means.

117. (previously presented) A system according to claim 116, wherein the Ports are specified in the form of regular expressions that select subsets of signals to be connected and define connection rules.

118. (currently amended) A system according to claim 105, wherein the ~~Configurator~~ configuration means is further configured to generate Transfer Components which are inserted on each side of the interface between servers, said Transfer Components comprising wires for inputs and registers for outputs.

Claims 119-129 (cancelled).

130. (New) A method for automatically generating a simulation model of a configuration of software simulation elements, comprising:

storing a plurality of software simulation elements, said plurality of software simulation elements provided with inter working connections so as to constitute the simulation model of an architecture, wherein the simulation model comprises models of integrated circuits for the realization of a machine that conforms to a functional specification of a configuration defined in a configuration definition file;

creating a simulation of wiring by executing stored regular expressions;

using the configuration definition file, a component and connection rule table, and a connection coherency rule table, wherein the component and connection rule table and the connection coherency rule table are written in a high level language, and the component and connection rule table describes properties of software components for simulating at least one of the integrated circuits;

instantiating components resulting from the configuration definition file; and

combining, via an HLL code generator, the parameters of the components with the connection rules of the component and connection rule table.

131. (New) A method according to claim l30, wherein the components comprise Active Components, Monitoring and Verification Blocks, Intermediate Blocks, System Blocks, and Global Blocks.

132. (New) A method according to claim 131, further comprising performing a conformity check of the connections by comparing an instance connection table with a table of coherency rules for the physical connections between the models chosen from the blocks constituting the model.

133. (New) A method according to claim 132, further comprising:

comparing the instance connection table to the connection coherency rule table to detect any incompatible connections between the ends of the connections between blocks; and

in cases where an incompatible connection is detected, specifying and adding an adapter component (Intermediate Block) to the instance connection table, said adapter component being inserted into the incompatible connection between the components.

134. (New) A method according to claim 133, wherein the component and connection rule table includes properties of the components and contains parameters common to all of the component types and exists in the form of a table distributed into one or more associative tables, and the entries being names designating all of the possible models for the same component.

135. (New) A method according to claim 134, wherein the associative tables are adapted to contain a description either in the form of parameter sets or in the form of references to procedures that generate the required values, and

wherein the entries of these associative tables comprise names designating all of the possible models for the same component, and form a character string containing predetermined special identifiers that are replaced by calculated values.

136. (New) A method according to claim 135, further comprising:

indicating, using at least three selectors, the instance to be used; and

transmitting the following selectors as parameters to a constructor of an HLL object:

a first selector indicating a current instance (item);

a second selector specifying the current instance connected to the other end of the port; and

a third selector indicating a composite instance corresponding to an active Component containing an observation port.

137. (New) A method according to claim 130, further comprising:

representing, by one or more connection coherency rule tables, the rules for interconnecting the components and for inserting intermediate components;

representing, by one or more component and connection rule tables, the system-level connection rules and the rules for generating connections between the signals; and

representing, by one or more source file formatting tables, the rules for generating instances of HLL objects.

138. (New) A method according to claim 130, further comprising:

uniquely identifying, via an HLL base class, each object instantiated and polling the configuration;

generating and automatically instantiating System Blocks;

using tables to associate the signals connected together under a unique name of the connecting wires; and

using a formatting table to generate hardware description language and HLL source files.

139. (New) A method according to claim 130, further comprising:

receiving, from an operator, a functional specification of the configuration in a high level language; and

completing the functional specification with the components in a language having a level lower than said high level language.

140. (New) A method according to claim 130, further comprising:

defining, using the following entries in a hash, a Component Type; and

correlating, using the following entries in the has, each Component Type to the hash, wherein said hash comprises the following:

a first entry comprising a name of a hardware description language (HDL) module of a component and a name of a corresponding source file; and

a second entry comprising a definition of a method for selecting the signals that are part of a Port, said definition comprising a set of entries indexed by a name of the Port; wherein

said method further includes associating each said Port name with a table of regular expressions and a pointer to a signal connection procedure that controls the application of the expressions to the names of the signals of the interface of the component.

141. (New) A method according to claim 140, wherein said Component Type comprises one or more Active Components having a generic structure that includes a containing Block that contains an HDL Block including an HDL description and a Block in HLL that provides access paths to HDL resources and a description of the containing block in the high level language;

wherein the set of signals of the HDL Block constitutes an interface of the containing Block, formed by Ports, which are arbitrary logical selections of the signals of an interface, and also formed by interface adapters which are the software parts that handle, in each Port, the two-way communication between the parts in high level language and those in hardware description language.

142. (New) A method according to claim 141, further comprising specifying the Ports in the form of regular expressions that select subsets of signals to be connected and define connection rules.

143. (New) A method according to claim 130, further comprising generating Transfer Components which are inserted on each side of the interface between servers, said Transfer Components comprising wires for inputs and registers for outputs.